CC4302 Sistemas Operativos – Tarea 6 – Semestre Primavera 2023 – Prof. Luis Mateu

Programe un sistema *batch* que permita solicitar la ejecución asíncrona de *jobs* por medio de la función *submitJob* y esperar hasta que un job termine con *waitJob*. Este sistema se ejecuta en una máquina sin sistema operativo y por lo tanto la única herramienta de sincronización disponible son los spinlocks. Los jobs se atenderán en un número fijo de cores con dedicación exclusiva a ejecutar jobs. Concretamente se pide implementar la siguiente API:

- *void initBatch( )* : Inicialice en esta función la variables globales como las 2 colas requeridas.
- *void cleanBatch( )* : Libere acá los recursos globales requeridos.
- *Job* submitJob(void* (*f)(void *ptr), void *ptr)* : Solicita la invocación de *(*f)(ptr)* en el sistema batch. Esta función no espera. Retorna de inmediato la dirección de un descriptor del job que será usado posteriormente al llamar a *waitJob*. Los jobs se asignan a los cores dedicados a atender jobs por orden de llegada.
- *void *waitJob(Job *job)* : Espera si es necesario a que termine el job especificado, entregando el resultado de la invocación de *f*. Con la invocación de *waitJob* se dice que el job está completo y el descriptor es liberado.
- *void batchServerFun( )* : La función que ejecuta cada uno de los cores dedicados a la ejecución de los jobs. Debe ejecutar un ciclo en donde (i) espera la llegada de un job si no hay ninguno en espera, (ii) ejecuta el job que lleva más tiempo en espera, y (iii) despierta al core que estaba en espera de ese job.

Metodología obligatoria:

- Para programar la sincronización requerida Ud. debe usar spinlocks. No puede usar otras herramientas de sincronización como mutex/condiciones o semáforos. Tampoco puede crear nuevos threads con *pthread_create*.
- Use una cola de los cores disponibles para ejecutar los jobs. Almacena las direcciones de spinlocks en donde la función *batchServerFun* espera la llegada de un job.
- Use una segunda cola para jobs esperando ejecución. Almacena las direcciones de los descriptores de job (tipo *struct job*). Defina la estructura del descriptor de job, incluyendo un spinlock para que *waitJob* pueda esperar hasta que el job se ejecute. Además incluya campos para *f*, *ptr* y el resultado de *f*.
- En la función *batchServerFun*, si la cola de jobs está vacía, cree un spinlock, agréguelo a la cola de cores y espere hasta que la llegada de un job lo despierte. Una vez que la cola no esté vacía, extraiga un job y ejecútelo. Al terminar despierte al core que esperaba ese job abriendo el spinlock incluido en el descriptor de job. Repita lo mismo hasta que reciba un job en donde la función *f* es NULL. En tal caso libere ese descriptor de job y retorne de la función *batchServerFun*.
- En *submitJob*, cree un descriptor de job con *malloc* y encólelo en la cola de jobs. Si la cola de cores no está vacía, extraiga un spinlock de esa cola para despertar un core que ejecute ese job.
- Todos los *submitJob* tendrán un *waitJob* asociado. En *waitJob* deberá liberar con *free* el descriptor del job. La excepción son las invocaciones de *submitJob* en donde *f* es NULL. En tal caso debe liberar el descriptor en la función *batchServerFun* porque no habrá *waitJob* asociado.
- No olvide garantizar la exclusión mutua con un spinlock global.

Prueba de la tarea: La verificación del funcionamiento correcto de su tarea se realizará primero implementando los spinlocks con mutex y condiciones, sin busy-waiting, y luego usando verdaderos spin-locks que sí esperan con busy-waiting y por lo tanto, se ocupa el 100% de la CPU. El primer método es útil para detectar dataraces.

Instrucciones

Descargue *t6.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros en el directorio *T6* para recibir instrucciones acerca del archivo en donde debe programar su solución (*batch.c*), cómo compilar, probar y depurar su solución, los requisitos que debe cumplir para aprobar la tarea y cómo entregar su tarea por U-cursos.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *batch.zip* generado por *make zip*. Este incluye *batch.c* y *resultados.txt*. Recuerde descargar el archivo que subió, descargar nuevamente los archivos adjuntos y volver a probar la tarea tal cual como la subió a U-cursos. Solo así estará seguro de no haber entregado archivos incorrectos. Se descuenta medio punto por día de atraso. No se consideran los días de receso, sábado, domingo o festivos.